Software Engineering  
G22.2440-001  

Session 8 – Sub-Topic 2  
Middleware JEE Reference Architecture  

Dr. Jean-Claude Franchitti  

New York University  
Computer Science Department  
Courant Institute of Mathematical Sciences
Middleware JEE Reference Architecture(s)
Executive Summary

- Identify current and proposed (i.e. new enterprise project) reference architectures
- Document architecture designs in order to establish enterprise-wide middleware standards
- Provides high-level view and examples of representative architectures
- Introduce a key reference architecture concept “the design pattern”
Glossary

- JEE - Java 2 Enterprise Edition
- EJB - Enterprise Java Bean
- MDB - Message-driven Bean
- BMP - Bean Managed Persistence
- CMT - Container Managed Transactions
- JMS - Java Messaging Service
- CU - Common Utilities
- WSAD – WebSphere Studio Application Development
Key Requirements

- Based on hardware direction, define software architecture standards for an application development centered on JEE technologies
- Facilitate repeatable designs in order to enhance programmer productivity
- Identify performance-oriented application constructs
- Minimum design complexity in order to reduce application maintenance cycles
Establishing Reference Architecture(s)

- A Reference Architecture is a starting point for architectural development.
- The use of tested Reference Architectures is an effective way to address non-functional requirements, by selecting existing architectures.
- A Reference Architecture is a set of design patterns proven for use in a particular business or technical context. These artifacts are usually harvested from previous projects.
- The key to a well defined Reference Architecture is well-defined patterns.

* Source: Reference Architecture from Rational RUP, 12/10/2003
…based on Design Patterns

- A pattern is a fully realized form, original, or model accepted or proposed for imitation...according to a dictionary.

- Patterns represent recurring solutions to software development problems within a particular context. Patterns and frameworks both facilitate reuse by capturing successful software development strategies. The primary difference is that frameworks focus on reuse of concrete designs, algorithms, and implementations in a particular programming language. In contrast, patterns focus on reuse of abstract designs and software micro-architectures.

- Using patterns in systems modeling helps in keeping design standardized. Typically, patterns are captured and documented in a sufficiently descriptive manner in UML, so that they can be referred for future use. Documenting patterns is one way that you can reuse and possibly share the information that you have learned about how it is best to solve a specific program design problem.*

* Source: Gamma, Helm, Johnson and Vlissides. Design Patterns. The seminal book on patterns.
This is a very simple design pattern for retrieving data from a database that includes two classes – SelectOrder and OrderRows.

- SelectOrder has behavior for binding with a DataSource and executing an SQL statement.
- OrderRows serves as a local container for the SQL result set.
- The classes for this pattern were generated using a WSAD wizard.
Example 2: Model-View-Controller

- Model-View-Controller pattern splits application into three distinct roles.
- Within the MVC Pattern there are, essentially, sub-architectural layers.
- For example, the Model layer can be further sub-divided into business and data access (synchronous and asynchronous) layers.
- Common Utilities include fundamental software architect design patterns for logging, exception handling, getting database connections, etc.
- Standard application patterns are being used within a number of enterprise applications, such as, Mainframe Migration, Membership, Fraud and Contact History.
- The pattern matrix on following page is used to identify the appropriate patterns based on an application processing style requirements.
### Possible Design Patterns

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Business</strong></td>
<td>- Front Controller implemented using servlets</td>
<td>- Front Controller</td>
<td>- Front Controller</td>
<td>- Front Controller</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Facade implemented as &lt;stateless&gt;&gt; Session EJB</td>
<td>- Singleton insures single object instance for Factory Pattern creation and used in CU JNDI lookup, log service and DB connection patterns</td>
<td>- Singleton insures single object instance for creation of CU JNDI lookup, log service and DB connection patterns</td>
<td>- Singleton insures single object instance for creation of CU JNDI lookup, log service and DB and MQ connection patterns</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- HTTPDispatcher similar functionality to Command Pattern (implemented in CU)</td>
<td>- Factory creates appropriate reentrant object, based on parameterizes request service name</td>
<td>- Command encapsulates request and response and executes object method based on parameterized service name</td>
<td>- Service Locator</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Service Locator (aka NDICacheRepository) JNDI look-up and caches references to EJB, JMS and DataSource services (implemented in CU)</td>
<td>- Identity Map insures that each object gets loaded once and only once by keeping loaded object in a HashMap.</td>
<td>- Service Locator</td>
<td>- LogService</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Integration (Data Access)</strong></td>
<td>- FastLane Reader 1 implemented using Java Bean accessed by servlet</td>
<td>- SQL Repository 1 using CU Constants Interface to contain SQL statements and DataSource references</td>
<td>- SQL Repository 2 using IBM .xst file format to contain SQL statements references</td>
<td>- FastLane Reader implemented using Java Bean or SQL Bean Pattern accessed by servlet</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Session Facade use methods on the EJB such as, insert(), update() and delete() for performing CMT</td>
<td>- Data Access Object decouple the data access from the either Servlet or EJB used in conjunction with Session Facade Pattern</td>
<td>- SQL Repository using CU Constants Interface to contain SQL statements and DataSource references</td>
<td>- SQL Repository using IBM .xst file format to contain SQL statements references</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Transfer Object transfer object is a serializable class that groups related attributes, forming a composite value</td>
<td>- FastLane Reader 3 implemented using SQLToXMLs or DAO Pattern</td>
<td>- FastLane Reader implemented using SQLToXML</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- SQL Repository 2 using IBM .xst file format to contain SQL statements references</td>
<td>- SQL Repository using IBM .xst file format to contain SQL statements references</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Session Management</strong></td>
<td>- Stateful Cookie uses a file to contain transient session state between J2S (Web Server) and browser</td>
<td>- Stateful Cookie</td>
<td>- Stateful Cookie</td>
<td>- Stateful Cookie</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(High-Availability)</td>
<td>- Memory-based Replication session state is multicast in-memory to a replica server within the cluster on another LPAR</td>
<td>- File-based Replication session state serialize to a file</td>
<td>- File-based Replication session state serialize to a file</td>
<td>- File-based Replication session state serialize to a file</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- DBMS Replication session state persisted to a DB</td>
<td>- DBMS Replication session state persisted to a DB</td>
<td>- DBMS Replication session state persisted to a DB</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Utilities</strong></td>
<td>- LogService application-bound system logging for debugging and information purposes (implemented in CU)</td>
<td>- LogService</td>
<td>- LogService</td>
<td>- LogService</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Exception provides consist error handling for enterprise applications (implemented in CU)</td>
<td>- Exception</td>
<td>- Exception</td>
<td>- Exception</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- DBConnectionFactory used to get a connection to a DBMS (implemented in CU)</td>
<td>- DBConnectionFactory</td>
<td>- DBConnectionFactory</td>
<td>- DBConnectionFactory</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Declarative Role-based Security</td>
<td>- Declarative Role-based Security</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- LogService</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Exception</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- LogService</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Declarative Role-based Security</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Exception</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- MQConnectionFactory used to retrieve a connection to MQSeries (implemented in CU)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Send and Receive Patterns One-way JMS pattern using a one-way operation. The sender sends a message to a destination and the receiver consumes it. No reply message is expected by the sender. (implemented in CU)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- LogService</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Exception</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Declarative Role-based Security</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
High-performance OLTP

Controller(s)
- Service Locator
- HTTPDispatcher
- Front Controller (Servlet)

Model
- Facade (Stateless Session EJB)
- Common Utilities (DBConnectionFactory, LogService, Exception)
- SQL Repository 1
- FastLane Reader 1
- JDBC

View
- HTML Page #1
  - Submit
  - Request
  - Cookie
- HTML Page #2
  - Response
  - Cookie
- XML/Fixed Length

IIS (Web Server)
Medium-performance OLTP

Controller(s)
- Service Locator
- Front Controller (Servlet)

Model
- Factory
- Transfer Object
- Command & Identity Map
- Common Utilities (DBConnectionFactory, LogService, Exception)
- Facade (Stateless Session EJB)
- FastLane Reader 2 *
- SQL Repository 2
- DAO
- JDBC
- DB

View
- HTML Page #1
  - Submit
  - Request
  - Cookie
- HTML Page #2
  - Response
  - Cookie
- XML/ XSL
  - Forward Response
  - Request/ Response

* See next slide for a UML view of the FastLane Reader 2
Sequence Diagram -- FastLane Reader

1: init()
2: InitialContext()
3: lookup(name)
4: QueryProperties()
5: getFile(relativeFilePath)
6: \load\n
7: doPost(request, response)
8: runQuery(request, response, queryoption)
9: ByteArrayOutputStream()
10: PrintWriter(out)
11: SQLToXMLds(qprop, ds)
12: setXMLWriter(printwriter)
13: setParameters(s)
14: execute()
15: getConnection()
16: setConnection()
17: finalize()
18: toString()
19: PrintWriter(out)
20: println(result)
21: close()

Response in XML

NOTE: Load template file for all SQL templates in this case for KS 04 and 34

Source: Lightweight Architecture Patterns
High-performance Data Inquiry

**View**

- HTML Page #1
- HTML Page #2
- Request
- Cookie
- Response
- Cookie

**Controller(s)**

- IIS (Web Server)
- Front Controller (Servlet)
- Service Locator

**Model**

- FastLane Reader 3
- SQL Repository 1
- DB
- Common Utilities (DBConnectionFactory, LogService, Exception)

**Controller(s)**

- Request
- Forward Response

**Model**

- Request
- JDBC
Complex Synchronous Data

**View**
- HTML Page #1
  - Submit
- HTML Page #2
  - Request
  - Response
  - Cookie
- XML/XSL
- Request/Response
- Forward Response

**Controller(s)**
- IIS (Web Server)
- Front Controller (Servlet)
- Service Locator

**Model**
- Factory
- Command & Identity Map
- Transfer Object
- Common Utilities (DBConnectionFactor, LogService, Exception)
- FastLane Reader 2
- JDBC

**Database**
- SQL Repository 2
- JDBC
- Java Stored Procedure
- DBFacade (Stateless Session EJB)
Batch Data Processing
Asynchronous High-performance

Controller(s)

Model

View

Controller(s)

Model

View
Appendix

- Lightweight architecture patterns exist and will continue to grow and be updated as patterns are harvested or new patterns are introduced. Lightweight Architecture patterns available at the WebSphere Best Practices site.
- User Interface Architecture available at the WebSphere Best Practices site.